Journaeceivedl of ICT, Design, Engineering and Technological Science (JITDETS)
VOL. 2, NO. 2, pp. 40-46, 2020
DOI: https://doi.org/10.33150/JITDETS-2.2.3

ORIGINAL CONTRIBUTION

Call-Center Virtual Assistant Using Natural Language Processing and Speech Recognition

Andrei Vasilateanu [1*], Razvan Ene [2]

[1, 2] Politehnica University of Bucharest, Bucharest, Romania

Abstract— **We propose a proof of concept that uses state-of-the-art cloud-computing technologies to lay those basic functionalities that we have envisioned as our solution. One of our objectives is to test available NLP and cloud platforms. We follow a case-study approach to the research, in which the prototype is tested in different scenarios, including short and long conversations, and the results are examined and discussed. In more detail, we involve nine speakers, with different accents and speaking styles, in 85 scripts and 3 real-life conversations. Results are promising, with good accuracies for short conversations and limitations due to the lack of domain-specific knowledge bases applicable to the call-centre work. Call centre assistance is one of the many domains of activity that Artificial Intelligence could enable. Enter Customer Recommended Interaction Software (CRIS). The idea of a virtual agent that can offer assistance during a live call or troubleshooting procedure has great potential. It can be used to unlock a great extent of advantages. The virtual agent presents a dashboard to the employee, with relevant facts about the conversation presented in real-time. The dashboard contains elements such as call-category based on detected keywords and sentiment analysis.**

Index Terms— **Serious Games, Education, Technical Communication, Development of Competencies**

Received: 23 September 2018; **Accepted:** 28 October 2018; **Published:** 28 December 2018

I. INTRODUCTION

This paper presents a prototype for a virtual, cloud-based, AI-enabled assistant that can operate in the confines of a call-center. To aid in the process of by-telephone troubleshooting and call analysis, the virtual agent breaks down the audio input from the interaction between the client and the operator and by doing so, it can offer guidance and share snippets of information (scorecards) concerning the user, his whereabouts, his client profile, call history, the initial problem classification, identified keywords, and sentiment analysis. The main feature is to provide the agent with an initial heads-up to better evaluate the problem and quickly start working on solving the client's issue.

We're developing a solution that involves having a virtual agent that listens to the conversation between customer and operator, gathers relevant data that can be collected from this interaction (audio input, text transcript, sentiment analysis, call duration, call silence, agent talk time, client talk time, keyword extraction, etc.) and produces both an initial output that the agent can look at when picking up the call as well as a post-call analysis which aims to provide a better understanding of the call-center's day-to-day activity by monitoring call specific Key Performance Indicators (KPIs).

An important milestone in the project development was the ability to process and transform audio input from over 500 live recordings into text format, with the help of Google's state-of-the-art speech recognition Application Program Interface (API) (i.e., Cloud Speech-to-Text) [1] and their latest Natural Language Processing (NLP) model (i.e., Natural Language) [2], also in the form of an REST API. All of our tests are centered around these recordings and their respective output.

In order to better simulate the conditions of a real-life call-center company, we focused on the telecommunications industry. This means that all of our recordings are centered around telecom specific problems, issues, and incidents (e.g., negative credit on SIM-card, incorrect billing, poor reception signal, and phone-related technical difficulties).

During the recording stage, 9 different speakers were involved, in order to better simulate different accents and speaking styles. We had 85 predefined scripts that were recorded individually by each speaker and 3 live, approximately 4 minutes long, real-life conversations, each having a different topic. The end goal consists of capturing different types of problems based around the idea that our user would get the chance to speak to the robot before getting redirected to a live agent.

By using these scripts and applying Google's speech recognition model, we tried to use basic NLP concepts, such as entity extraction and sentiment analysis. Entity extraction is paramount in processing and evaluating keywords, as it is our primary tool to understand what the person is trying to convey.

Another key-development stage was the ability to perform entity extraction and sentiment analysis on each transcript. Following this stage will be the creation of a predefined knowledge base, in which the context

[*] Corresponding author: Andrei Vasilateanu
[†] Email: andraevs@gmail.com

of the conversation will be detected by our virtual agent.

This knowledge base would consist of a keyword and key-phrase dictionary, which will be used to asses the context of an individual utterance (i.e., if we want to process the 15-30 second window in which the client initially states his call reason) or the general context of an entire agent-client conversation. By extracting relevant entities within the audio file, after having it transformed into text via Google's speech recognition API, we can compare them to our existing ones and determine the heat-map of detected categories.

Correctly identifying call categories would be the first step in automating the task that the call-center agent would be doing at the end of the call manually. The agent needs to correctly assess the nature of the call so that all preceding KPIs are related to its context (e.g., "Information Request", "Technical Issues", "Billing and Invoices", etc.).

II. LITERATURE REVIEW

A highly researched sub-topic of artificial intelligence mainly concerned with the processing of human-computer interactions by means of language interactions, NLP is trying to process and analyze any amount of data that consists of human to computer interaction and output some form of result that is indicative of meaning, context, or any other language-related procedure. NLP is comprised of several techniques; we describe those relevant to our project.

Sentiment analysis inspects the given text and identifies the prevailing emotional opinion within the text, especially to determine a speaker's attitude as positive, negative, or neutral. Entity analysis inspects the inputted text for known entities (proper nouns such as public figures, landmarks, and so on, common nouns such as restaurant, stadium, and so on.) and returns information about those entities. Entity sentiment analysis inspects the given text for known entities (proper nouns and common nouns), returns information about those entities, and identifies the prevailing emotional opinion of the entity within the text, especially to determine a speaker's attitude toward the entity as positive, negative, or neutral. Syntactic analysis extracts linguistic information, breaking up the given text into a series of sentences and tokens (generally, word boundaries), providing further analysis on those tokens. Content classification analyzes text content and returns a content category for the content. This can be customized to fit specific needs. In the case of CRIS, we intend to categorize problems that the users are having into the support categories that the company which owns the call center has (Technical Issues, Billing Issues, Legal Issues, or any other issues categories SAM will be tailored to address).

Name entity extraction also comes into play, often used to analyze the conversation and decide on what path an automation system should take the course of action, given some entity triggers that it searches for [3, 4, 5, 6, 7].

Another relevant topic for our project is speech-recognition. The term speech recognition refers to the study of developing methodologies and practices that enable the recognition of a translation of spoken language into text format by computers. No doubt, an interdisciplinary field of study, it combines elements from linguistics, computer science and electrical engineering to form a solution capable of doing what was mentioned earlier. The speech to text aspect of this project was achieved by using Google's Cloud-based Speech API, which uses state of the art, powerful, cutting edge machine learning models that are tried, tested, and backed-up by a substantial amount of data in order for them to work properly and accurately [8, 9, 10].

Regarding the actual application of our framework–supporting call-center operations, the literature has few results. We mention some items in related fields. In [11] the authors approach sentiment analysis for social media also using models based on ontologies. Our initial results show that ontologies are not necessary for sentiment analysis. Also, the study uses a pipe-and-filter architecture, not always applicable in real-time applications such as ours. In [12] the research focuses on how dialect can affect speech recognition. It is a relevant topic for our research. However, it is not planned for the first iterations of the project. Our research has only approached the English language. However, it is clear that we must take into consideration variations that can appear in other languages, especially for transcribing speeches [13]. In addition to sentiment analysis, more complex approaches can be used for customer satisfaction, such as those presented in [14].

To conclude, while NLP and speech recognition are research subjects with a lot of traction, their actual application in call-center settings has not been explored enough. It is possible though to use research in related domains, of course, translated to the specifics of call-centers.

III. METHODOLOGY

A. Introducing the Google Cloud Platform

Google's public cloud service is a well-known standard in the industry and prides itself as being build on the same infrastructure as any other Google product (e.g., Google Search, G-Mail, Google Drive). This cloud is built on top of one of the largest, most technologically advanced computer networks currently running in the world. It's comprised of thousands of fiber-optic, high-speed cables, creating kilometers upon kilometers of modern infrastructure. Google's main achievement in creating this network consists of being one of the few companies currently owning a private fiber-optic cable through the Pacific Ocean.

Our main reasoning behind choosing Google Cloud Platform (GCP) is its high level of availability in terms of up-time, its seamless integration, in the form of REST APIs, with multiple client-libraries (e.g., Python, Java, JavaScript, HTML5/REST), well-written documentation and examples, as well as some of the best artificial intelligence and machine learning models available out-of-the-box (e.g., Cloud Speech, Natural Language, AutoML) [1, 2].

As far as our project is concerned, we opted for the following client libraries compatible with the Java SE8 programming language:

- Cloud Speech-to-text API: Google's speech recognition solution for over 120 different languages. This API enables developers to apply neural network models (the same that Google uses for their Assistant and voice command solutions) to audio input and output the corresponding transcript. It can process real-time or prerecorded audio files.
- Cloud Storage: The go-to GCP solution for storing data (buckets, blobs, objects, and files). The API enabled us to reliably store in a geo-redundant way all of our logs, audio files, and transcripts following our implementation. All the files are stored locally, and are readily available for access, via a unique URL.
- Natural Language: By using machine learning to interpret the meaning and extract structure from the text, we were able to analyze the output from over 500 live recordings. NLP will be the final interface between the machine, which tries to find meaning within the input, and the agent, trying to read and analyze the output to better understand the client's issue

B. Design and Implementation

The virtual agent is designed to offer a 15 to 30-second window in which the caller would state his issue and thus provide the first level of context and information concerning his problem. Using this initial audio input will help CRIS to form a first view and after processing the data will begin to feed that information to the agent's dashboard. The operator will thus be able to visualize what the robot believes to be the issue present and act accordingly.

What we also want to achieve is to store all conversations that took place and all the audio files used to develop an analytics-based output. We aim to produce relevant reports, based around identified KPIs, relevant to a call-center domain of activity (call duration, customer feedback, etc.)

The primary objective of this proof of concept is to show that audio files can be captured by our algorithm, transformed into text by applying speech recognition and using that text to extract all relevant information that can be used to form a more in-depth understanding of the client's issue. Some relevant metrics we want to capture are standard call metrics and KPIs (call duration, call silence, agent talk time, client talk time, etc.) as well as captured entities (keywords and key-phrases used to identify the topic of the conversation and to establish the recommended course of action) and sentiment analysis (picking up feedback provided by the client, sentence by sentence, and using it to form of experimental, automated feedback generation solution).

C. High-level Overview

When designing our solution architecture, we wanted to successfully accomplish 3 things: processing audio files, transforming audio into text, and applying natural language processing to extract entities.

We've opted to use the Java API in order to access Google's cloud solutions, and following this, we developed an easy-to-use CLI-style Java application that demonstrates the basic capabilities of 3 different APIs: Cloud Speech-to-Text, Natural Language, and Cloud Storage.

While still being in a proof of concept stage, the application clearly met its goal, and we've achieved satisfactory output. Meaning that we've successfully analyzed all of our recordings after storing them in the cloud repository, produced for each one of them a resulting transcription, and gathering relevant entity-related information from that text.

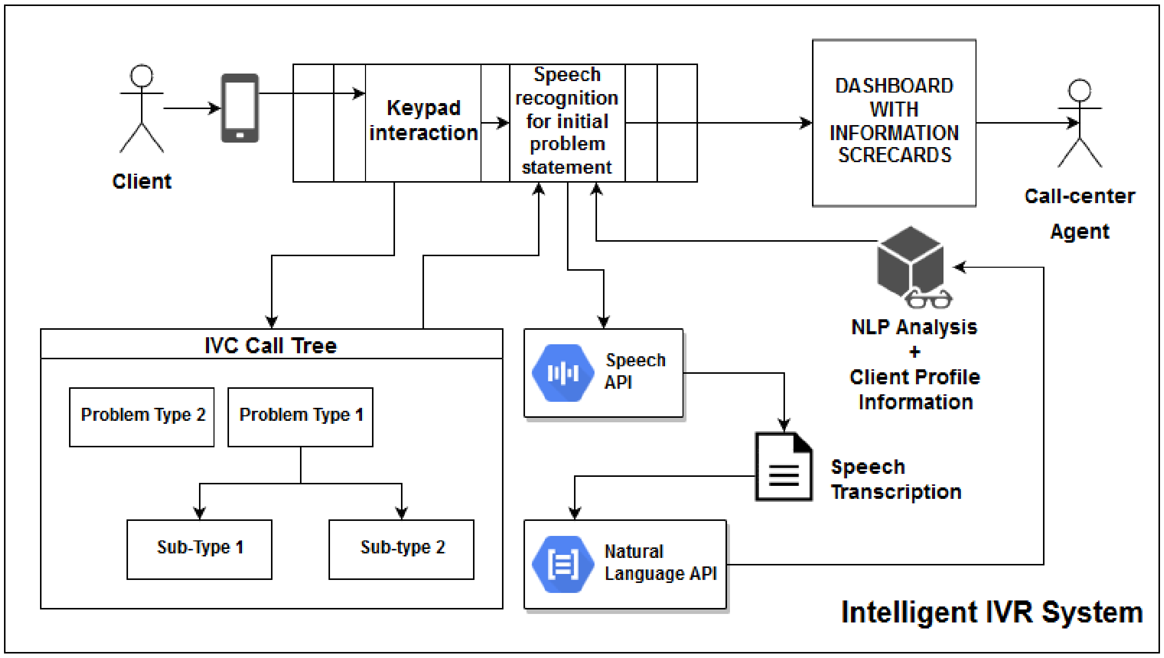

Fig. 1. Proposed high-level architecture

Let us detail recordings for better understanding. We've split our efforts into two large categories:

- Long audio recordings (greater than 1 minute in length)
- Short audio recordings (less than 1 minute in length)

Long audio recordings aimed to capture real-life conversations between client and operator. We've gained access to anonymous transcriptions and were able to translate them into English. All recordings were done by 9 different speakers, each taking turns as client and operator.

The first call was related to the configuration of APN (Access point Names). He was helped by the operator to configure it properly and felt rather satisfied with the service provided.

The second call splits its topic mid-way through the call. A lady is requesting assistance with her daughter-in-law's mobile phone data connection, and after solving this issue, she asked the operator to provide her with some information about her phone's warranty. Both issues were solved by the operator.

The third call has a customer calling to request assistance with his router internet connection. The SIM-card router was acting problematically since a recent power outage, but that was not the cause of the problem. Following some investigation done by the agent, it was revealed that the client requested his subscription cancellation himself and forgot about it. The agent was unable to solve the issue, and the client was left to decide whether he opted for a new contract or not.

The 85 short audio files were built using fictitious problem statements using the following categories as guidance:

- Changing mobile data provider.
- Subscription cancellation information request.
- Subscription upgrade or downgrade.
- The client presenting himself and his problem.
- Issues with phone signal.
- Unexpected costs in monthly bills and invoices.
- Random inquires about telecommunication services.
- Blocking phone numbers.

The scope of these short recordings was to emulate the use case in which the client would be able to state his issue before starting his call with the agent. This can be used to prepare the aforementioned dashboard for the operator, revealing information scorecards to help him visualize the problem correctly.

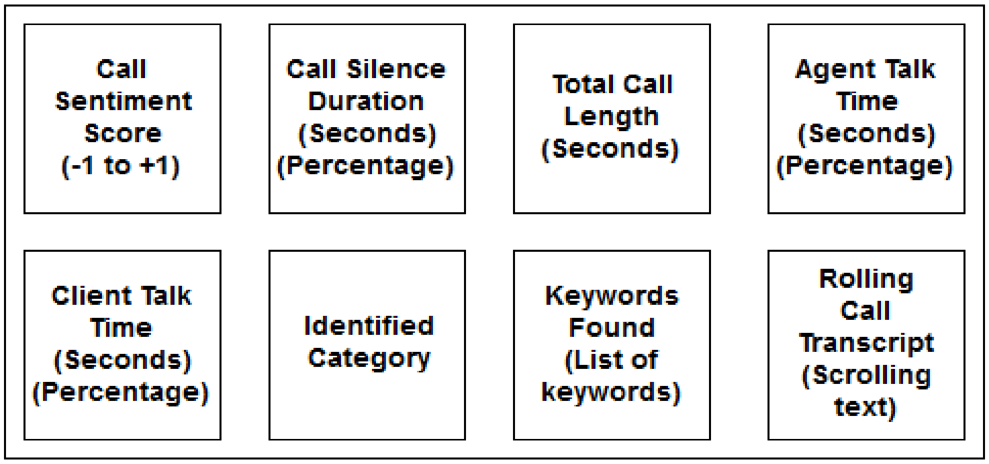
Call Sentiment Score (-1 to +1)
Call Silence Duration (Seconds) (Percentage)
Total Call Length (Seconds)
Agent Talk Time (Seconds) (Percentage)
Client Talk Time (Seconds) (Percentage)
Identified Category
Keywords Found (List of keywords)
Rolling Call Transcript (Scrolling text)

Fig. 2. Mock-up for information scorecard display

All recordings were performed using a smartphone microphone, a 16 kHz audio sample rate, FLAC encoding, and "en-US" language code for optimal performance and obtaining the best recognition values from Google's model.

The first step was to record all of our conversations and upload them to Google Cloud Storage in order to set up our recordings library. In order to split the workload of the recognizer, we stored them in 2 different buckets, one for long audio files and one for short audio files.

Files were stored redundantly on the cloud and accessed by means of a unique URI, specific to this cloud storage solution. This was the only point in our solution where we didn't opt for an end-to-end, server-less solution, as the files had to be manually uploaded to the storage bucket designated.

The URI is indexed by using the Cloud Storage API, where we could actively search using the filename itself, and the API would automatically generate the corresponding URI. This is done to "pull" the audio file and feed it to the recognizer.

To successfully run the demo, a user would have to type in each corresponding option and follow a step-by-step flow to generate the output desired. The first step is to successfully initialize both the audio recognizer as well as the cloud storage agent. Both of these objects would run in the software's background processes, and silently wait for incoming requests (e.g., retrieving a file or processing an incoming audio conversation).

D. Cloud Storage

Now that both the audio and storage clients have been successfully enabled by using the right API calls, we can start and query the storage system to extract a desired audio file.

Searching is done by typing in the desired file name. The software returns the valid selection, and the user is notified that he has selected a new active recording. With the active recording selected, he can then ask the software to produce the audio transcription.

Every action done within the session is stored in a log-file, and all output results are saved in JSON format. This step is to ensure that once we have our desired output, we can use it for later processing and following more development stages, building advanced analytic outputs in a scorecard-type display interface. Once the file has been successfully selected and stored within the local client session, we can start to produce some output using it.

The user now needs to use the console to input the next step, which is to generate the URI corresponding to his active storage selection. This fact tells the speech recognition client what file to process. This active URI will be shown as being his active selection until the user decides to change his active recording, in which case he needs to re-generate his URI.

E. Cloud Speech-to-Text

When the user has an active recording selected and a URI generated for him, he can call the speech recognizer and visualize the resulting transcription output.

One very important reason that resulted in us opting for Google's Speech-to-Text API is its superior accuracy compared to other models as well as the ability to produce automatic punctuation based on detected user intonation. Punctuation is the key to our next step, analyzing the output using natural language processing techniques. Once again, an API was used to produce the results; this time, it was the Natural Language API.

F. Natural Language Processing

The Natural Language API produces 2 distinct outputs that we're interested in: entity extraction and sentiment analysis.

Entity extraction is a technique that inspects a given text for known nouns (or otherwise known as "nominals" in the field of natural language processing) and returns information about them. A good rule is that if something can be interpreted as being a noun, then it also can be identified as being an entity within the text.

Nouns can be objects that the user is experiencing an issue with, the type of information that he is requesting from the call-center operator, or any other important concept we're likely to be interested in. Here is where we extract our keywords and key phrases (entities are not limited to one word only).

For our demo, we opted for Google's predefined model. While not the optimal solution for a production-level application, it was able to capture all the possible entities within any given chunk of text (assuming no inherent errors within the transcription are present).

In the near future, opting for a custom model that's tailored to detect entities that we're particularly interested in (i.e., having a model for a specific call-center domain of activity, such as emergency or IT-related) will boost our results and get us closer to our end solution.

Entity extraction also comes along with salience scoring. This concept is concerned with portraying the relevance of entities by assigning a score to each of them. Salience is still an experimental feature of the API, and it doesn't always portray what we humans see as being the most relevant words within a phrase. Salience was kept, in our case, for testing purposes. This scoring can be used in conjunction with a different metric (frequency of found entity within the paragraph, word-complexity of the entity, etc.) to produce a more refined result once we obtain more experience in working with the API itself. Salience scores range from 0 (less important entities) to 1 (highly important entities).

Sentiment analysis is more difficult to integrate at this point as it tries to portray objectively something subjective by nature, the emotional inclination of a given text. Saying if something is either positive or negative in nature is a complicated task but holds great potential in unlocking a highly desired aspect within modern call-center environments: automatic feedback generation from the customer.

This emotional analysis integrated within the Natural Language API has two metrics attached: sentiment score and magnitude score. Sentiment score is applied either to the entire text or to individual sentences (we've opted for both cases when building our proof of concept). It tries to detect the emotional leaning of the text. This score ranges from negative 1 to positive 1. It's a rough estimate with which we guess the emotional inclination of our call-center client. Sentiment score is a normalized value, meaning that given a long enough text, positive and negative expressions will most likely cancel each other out (when working with short phrases and individual sentences, this is rarely the case).

Magnitude score indicates the overall strength of emotion (both positive and negative). It is not normalized and ranges from 0 to positive infinity (each detected emotion contributes to the overall score of the text). Given this fact, larger chunks of text tend to have a larger magnitude score. Magnitude is effectively used to differentiate between similar emotional scores (two identical sentiment scores can be sorted by using the magnitude attached to them).

We've extracted all relevant entities, given our limited dataset. When handling conversation between agent and client, we opted for the following solution: splitting the NLP analysis into three distinct categories. The first is bulk analysis. What this means is that we took the entire transcript (agent and client together) and fed it to the Natural Language API model. The second approach was to split what the agent said and what the client said in two different text entries and then send it to the model for output. Finally, we opted to separate the conversation into individual sentences to achieve the best level of granularity.

IV. RESULTS AND CONCLUSION

The goal of this proof of concept was to propose a solution that will evolve into an "intelligent" call-center assistant.

This is achieved by observing the two different use cases put in place for our virtual agent architecture. The first is the pre-call phase, in which we prompt the user to state his problem during his waiting time (minimizing waste in the process and optimizing the call-center experience for both sides). This initial problem statement will be used (with the help of voice recognition and natural language processing) to extract data and build the initial dashboard which will be shown to the agent when he initiates the call with the client.

The dashboard will try to capture all relevant metrics when it comes to analyzing the call quality as well as adding our own ones (e.g., list of detected keywords, category classification, and sentiment analysis). By displaying these snippets of information in a scorecard-like manner, we can help speed up the troubleshooting procedure handled by the operator and capture relevant metrics from the call at the same time. These metrics will be analyzed post-call to try and build a better picture of the activity present within the call-center (i.e., if the client is satisfied overall with the services presented, if the agent is correctly identifying the nature of the call in due time, and if the virtual agent is correctly identifying the data it's designed to).

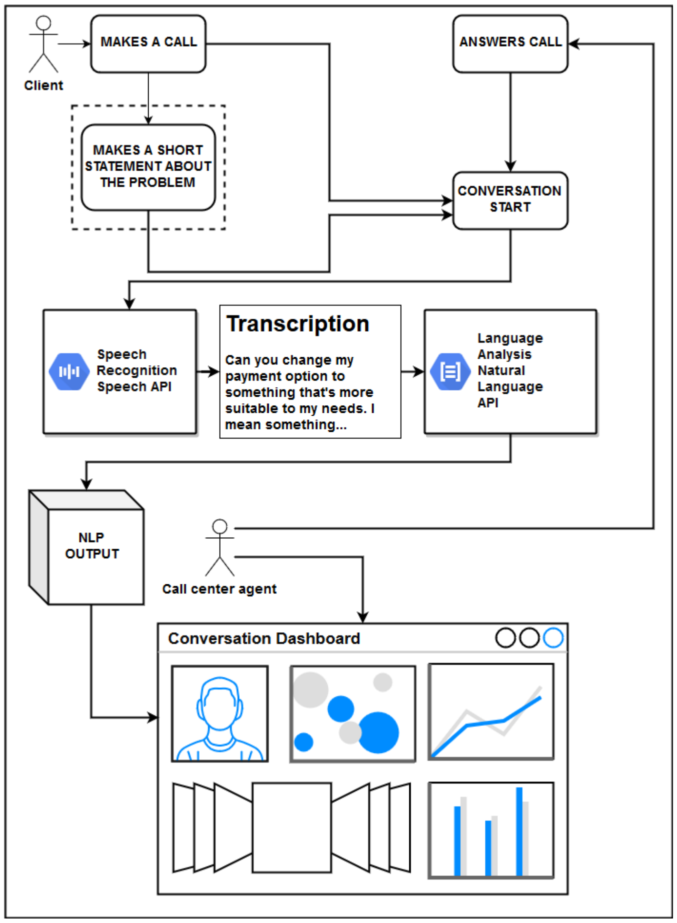
Client
MAKES A CALL
ANSWERS CALL
MAKES A SHORT STATEMENT ABOUT THE PROBLEM
CONVERSATION START
Speech Recognition Speech API
Transcription
Can you change my payment option to something that's more suitable to my needs. I mean something...
Language Analysis Natural Language API
NLP OUTPUT
Call center agent
Conversation Dashboard

Fig. 3. CRIS workflow

Regarding the results on both short and long audio files from a speech recognition perspective, it appears that the API is more suitable out of the box for handling short utterances (remember that it's the same technology that powers Google Assistant, which is concerned with handling commands via voice interaction), which means audio files shorter than 1 minute in total duration. This enables us to achieve better results with our first use case (capturing the client's intent before routing him to an operator).

For long audio files, speech transcription is in-between as far as the accuracy of detected words and punctuation is concerned (relative to short audio files, that is). It is likely due to the lack of audio recordings on the longer side of the spectrum (only 30 or so recordings of conversation were provided during this stage) and incorrect configuration on our part. The lack of real-life conversations (taken from call-centers) is something we need to address in the following stages of development. We're looking into solutions to gather more and more data, thus aiming at improving our speech detection accuracy.

Speech recognition comes enabled with data logging, which means that the recognition requests we are using to communicate with the cloud API are then used to retrain the model for better accuracy. This enables us to increase the performance of the model, given a sufficient pool of audio recordings available. Keep also in mind that speech recognition was done with mostly default configuration values for the sake of having a functional demo during this stage of development. As we experiment more with the technologies chosen, we will arrive at even better results.

Concerning the Natural Language API, taking into consideration the fact that this too was used with mostly default values for the sake of completion and demonstration purposes, we can safely state that it is close to our desired outcome.

Entity extraction is functional and provides us with a clear view of all the keywords and key-phrases detected within text inputs. The entities are what will help us to break down the meaning of what both the client and the agent are trying to say and use it to generate automatic classification.

To summarize, the main weaknesses of the study are the small number of long audio files and the lack of specific knowledge bases for the call center domains.

V. DIRECTION FOR FUTURE RESEARCH

The next step is choosing a couple of existing knowledge bases for call-center-specific activity domain keywords (i.e., a knowledge base for each identified call nature from a business stand-point). It means that we will limit the scope of the application so that it serves a specific call-center domain (emergency calls, telecommunications, sales, IT-support, etc.) and build a knowledge base that will be used to clearly pinpoint the nature of the client-agent conversation. We would compare the existing entities found within the text with our predefined knowledge base and perform an additional scoring to assess the correct category of any call (this will also enable us to detect multiple categories, as we will be able to display all scores from a percentage perspective).

Once we have these preexisting dictionaries, we will be able to start evaluating the accuracy of call categorization done by the virtual agent.

We need to decide on the scorecard metrics that are more relevant to our use cases and find a way to use the agent's feedback to improve the virtual assistant's performance. We're imagining the agent being able to correct the robot by assessing its output. Given enough time and data, we can improve the performance and accuracy of our solution using this closed feedback loop system.

Declaration of Competing Interest

The authors declare that there is no conflict of interest.

References

[1] Google Cloud. Speech-to-text documentation. [Online]. Available: https://bit.ly/31AbQ9z

[2] Google Cloud. Cloud natural language API documentation. [Online]. Available: https://bit.ly/2SvUE0y

[3] Y. Goldberg, ``A primer on neural network models for natural language processing,'' *Journal of Artificial Intelligence Research*, vol. 57, pp. 345-420, 2016. doi: https://doi.org/10.1613/jair.4992

[4] D. Nadeau and S. Sekine, ``A survey of named entity recognition and classification,'' *Lingvisticæ Investigationes. International Journal of Linguistics and Language Resources*, vol. 30, no. 1, pp. 3-26, 2007. doi: https://doi.org/10.1075/li.30.1.03nad

[5] X. Liu, S. Zhang, F. Wei, and M. Zhou, ``Recognizing named entities in tweets,'' in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies,* Portland, Oregon, 2011, p. 359–367.

[6] M. Mintz, S. Bills, R. Snow, and D. Jurafsky, ``Distant supervision for relation extraction without labeled data,'' in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - ACL-IJCNLP '09,* Suntec, Singapore, 2009, p. 1003–1011.

[7] N. A. Zainuddin, I. Norhuda, I. S. Adeib, A. N. Mustapa, and S. H. Sarijo, ``Artificial neural network modeling ginger rhizome extracted using Rapid Expansion Super-Critical Solution (RESS) method,'' *Journal of Advances in Technology and Engineering Research*, vol. 1, no. 1, pp. 1-15, 2015. doi: https://doi.org/10.20474/jater-4.2.5

[8] M. Pinola. (2011) Speech recognition through the decades: How we ended up with siri. [Online]. Available: https://bit.ly/2S6GtAj

[9] G. Pirani, *Advanced Algorithms and Architectures for Speech Understanding*. Heidelberg, Germany: Springer Verlag, 1990.

[10] E. B. Nejad and R. A. Poorsabzevari, ``A new method of winner determination for economic resource allocation in cloud computing systems,'' *Journal of Advances in Technology and Engineering Research*, vol. 2, no. 2, pp. 12-17, 2016. doi: https://doi.org/10.20474/-jater2.1.3

[11] P. Thakor and S. Sasi, ``Ontology-based sentiment analysis process for social media content,'' *Procedia Computer Science*, vol. 53, pp. 199-207, 2015. doi: https://doi.org/10.1016/j.procs.2015.07.295

[12] J. H. Hansen and G. Liu, ``Unsupervised accent classification for deep data fusion of accent and language information,'' *Speech Communication*, vol. 78, pp. 19-33, 2016. doi: https://doi.org/10.1016/j.specom.2015.12.004

[13] R. Lileikytė, L. Lamel, J.-L. Gauvain, and A. Gorin, ``Conversational telephone speech recognition for Lithuanian,'' *Computer Speech & Language*, vol. 49, pp. 71-82, 2018. doi: https://doi.org/10.1016/j.csl.2017.11.005

[14] T. El-Diraby, A. Shalaby, and M. Hosseini, ``Linking social, semantic and sentiment analyses to support modeling transit customers' satisfaction: Towards formal study of opinion dynamics,'' *Sustainable Cities and Society*, vol. 49, p. 101578, 2019. doi: https://doi.org/10.1016/j.scs.2019.101578